Dynamic Reconfiguration for Management of Radiation-Induced Faults in FPGAs

Maya Gokhale, Paul Graham, Michael Wirthlin, D. Eric Johnson, and Nathaniel Rollins

***Abstract*— This paper describes novel methods of exploiting the partial, dynamic reconfiguration capabilities of Xilinx Virtex V1000 FPGAs to manage single-event upset (SEU) faults due to radiation in space environments. The on-orbit fault detection scheme uses radiation-hardened reconfiguration controllers to continuously monitor the configuration bitstreams of 9 Virtex FPGAs and to correct errors by partial, dynamic reconfiguration of the FPGAs while they continue to execute. To study the SEU impact on our signal processing applications, we use a novel fault injection technique to corrupt configuration bits, thereby simulating SEU faults. By using dynamic reconfiguration, we can run the corrupted designs directly on the FPGA hardware, giving many orders of magnitude speed-up over purely software techniques. The fault injection method has been validated against proton beam testing, showing 97.6% agreement. Our work highlights the benefits of dynamic reconfiguration for space-based reconfigurable computing.**

***Index Terms*— radiation effects, SEUs, FPGAs, proton accelerator, half-latches.**

I. Introduction

Field Programmable Gate Arrays (FPGAs) offer significant advantage over microprocessors for space missions, which are characterized by demanding schedules, low budgets, and low volume. SRAM-based (as opposed to anti-fuse) FPGAs are especially appealing due to their in-situ reprogrammability and high performance for signal processing tasks. However, the use of commercial SRAM-based FPGAs in satellites and spacecraft presents unique challenges in the presence of the space radiation environment. Heavy ion testing [1] has shown that Xilinx Virtex XQVR300 SRAM-based FPGAs are single-event latchup (SEL) immune to up to a linear energy transfer (LET) of 125 MeV-cm^2/mg, but are sensitive to single-event upsets (SEUs) at an average threshold LET of 1.2 MeV-cm^2/mg with an average saturation cross-section of 8.0×10^{-8} cm^2. This means that in a Low Earth Orbit (LEO), the nine-FPGA system we have built (Figure 1) can be expected to experience radiation-induced upsets 1.2 times/hour in low radiation zones and 9.6 times/hour when there are solar flares.

While it is highly desirable to exploit dense, dynamically reprogrammable commercial SRAM-based FPGAs rather than radiation-hardened anti-fuse parts with 1/10 the capacity, it is clear that fault detection and mitigation must be addressed before these FPGAs can be deployed on satellites or spacecraft for compute-intensive applications.

In this paper, we describe the crucial role of dynamic reconfiguration in detecting and correcting SEU-induced transient faults as well as detecting and isolating more serious permanent faults. Our system has been built and will be launched as an experimental payload in 2006 [2]. The system uses dynamic readback and reconfiguration techniques to detect and correct SEU faults in Xilinx Virtex V1000 FPGAs. Partial reconfiguration can be used to detect and isolate permanent faults. In addition, we use a novel fault injection method based on dynamic reconfiguration to characterize SEU-induced configuration bitstream faults and to study the effects of these faults on the execution of our application set. This fault simulation technique has demonstrated a 97.6% correlation with the results found during a proton beam radiation study [3].

Manuscript received July 2, 2004. This work was supported by the Department of Energy through the Deployable Adaptive Processing Systems project at Los Alamos National Laboratory.

M. Gokhale and P. Graham are with Los Alamos National Laboratory.

D. E. Johnson, N. Rollins, and M. Wirthlin are with Brigham Young University.

Document Release Number: LA-UR-04-4553

II. Fault detection and correction in a space-based reconfigurable computer

Our reconfigurable computer will serve as space-based reconfigurable radio. The chassis shown in Figure 1 is based on the IEEE SEM-E standard [4]. This standard defines the mechanical environment for a collection of modules plus connecting backplane. In this assembly, each module consists of two printed circuit boards bonded to a metal core. The core dissipates heat into the spacecraft thermal management system. A radiation-hardened microprocessor, a 30-MHz RAD6000, controls the system and coordinates communication between the spacecraft and experimental payload. Other system components (see Figure 2) include non-volatile memory modules, an A/D module receiving input from the radio, and the FPGA compute modules (labelled “RCC” in the diagram).

The EEPROM provides 1MB of nonvolatile storage for the operating system and microprocessor application code space. The other printed circuit board on the EEPROM module holds the spacecraft interface for 10Mbit communication between the payload and ground station. The interface is used to send commands to the payload, upload configurations for the FPGAs, query state of health, and retrieve experimental data.

The 16MB flash memory module stores more than twenty configuration bit streams for the Xilinx FPGAs (without compression). Error control coding (ECC) is used to mitigate SEUs that might occur while the memory is being accessed. The EEPROM packaged with the FLASH memory is intended for additional application configurations.

The A/D modules sample two 40MHz RF channels at an IF

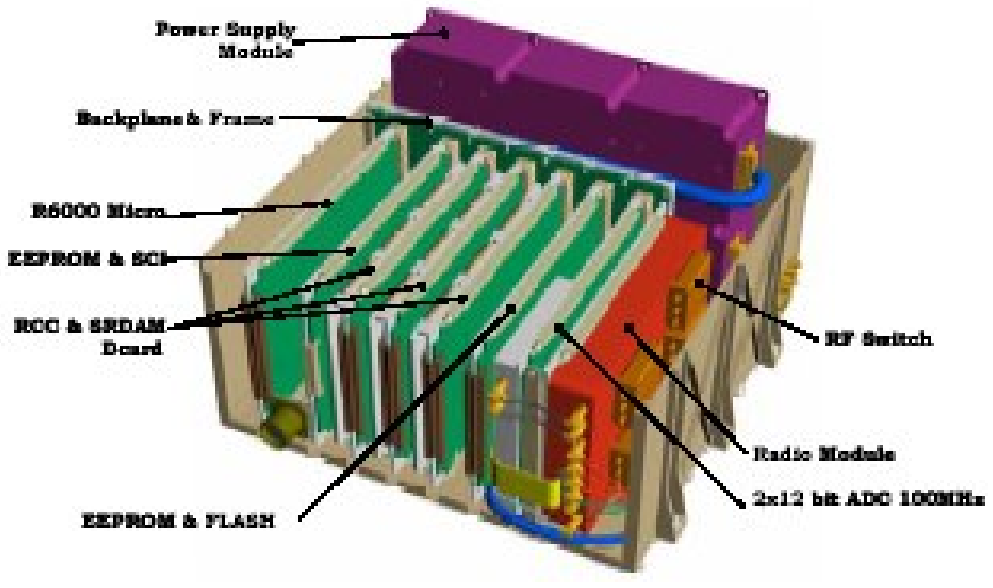


Fig. 1. Chassis of Space-Based Reconfigurable Radio

of 55-95MHz at 100MHz with 12-bit resolution. The digital IF is transmitted to the network of FPGA modules for processing.

The FPGA modules are used to process the Intermediate Frequency (IF) for ionospheric and lightning studies. The objective is to detect and measure impulsive events that might occur in a complex background, and to group together those events that coincide geographically.

The signal processing is performed on three compute boards containing a total of nine Virtex V1000s and 288 MB of SDRAM organized in 8 M $\times$ 32-bit modules. The Virtex FPGAs in our system are radiation-tolerant XQVR Virtex FPGAs. While these FPGAs uses the mask sets of existing commercial devices, they are fabricated on epitaxial silicon wafers to provide SEL immunity.

A. Detecting and Correcting SEU-Induced Transient Faults

Figure 3 shows one board of the reconfigurable computer. The FPGAs are organized in a ring. All the FPGAs have identical pinout, so that the same configuration bit stream may be loaded on any FPGA. The FPDP channels provide high-speed paths to pass data among the FPGA boards. These busses run at 50MHz with 32-bit data, giving at total bandwidth of 200MB/s. The Actel FPGA controller is a radiation-hardened anti-fuse FPGA that provides an interface to the microprocessor. This device is used to load configurations onto the FPGA. As importantly, the Actel serves as fault manager, providing watchdog monitoring of the FPGAs. The Actel scans each Xilinx FPGA for SEU faults by continuously reading the FPGAs' configuration bitstreams and calculating a cyclic redundancy check (CRC) for each frame of each configuration. The frame is the smallest granularity of reconfiguration available on the Xilinx parts [5]. The calculated CRC is then compared with a codebook of stored CRCs. The stored CRCs

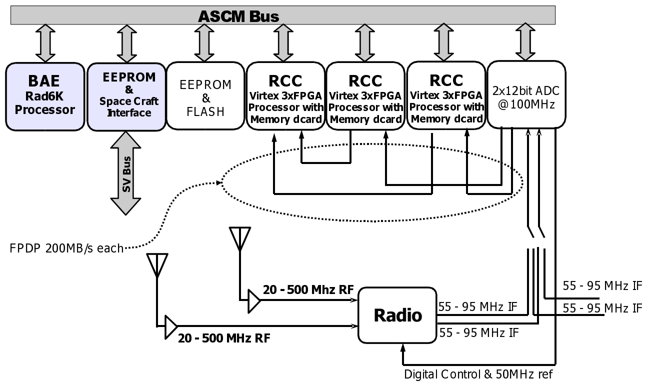

ASCM Bus
BAE Rad6K Processor
EEPROM & Space Craft Interface
EEPROM & FLASH
RCC Virtex 3xFPGA Processor with Memory dcard
RCC Virtex 3xFPGA Processor with Memory dcard
RCC Virtex 3xFPGA Processor with Memory dcard
2x12bit ADC @100MHz
SV Bus
FPDP 200MB/s each
20 - 500 Mhz RF
20 - 500 MHz RF
Radio
55 - 95 MHz IF
55 - 95 MHz IF
55 - 95 MHz IF
55 - 95 MHz IF
Digital Control & 50MHz ref

Fig. 2. Reconfigurable Radio Overview

are loaded from the FLASH module via the microprocessor and are kept in local SRAM.

Using the Virtex SelectMAP interface, each configuration is read every 180 ms while the FPGA is operating. There is no interruption of service required to perform readback. If an error is found, the microprocessor is interrupted and notified of the specific device and frame that was corrupted. This information is stored and later relayed back to the ground station, contributing to the "State-of-Health" record of the subsystem. The microprocessor fetches the original frame bitstream segment from FLASH, partially reconfigures the device to restore that frame (156 bytes for the XQVR 1000), and then resets the system. Figure 4 shows the fault detection and mitigation process.

The system also allows for artificial insertion of SEUs into the Virtex parts using the microprocessor to partially configure

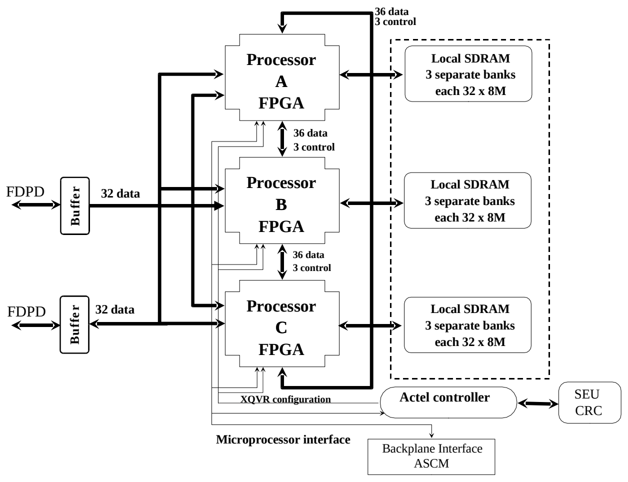


Fig. 3. FPGA board with Configuration Manager

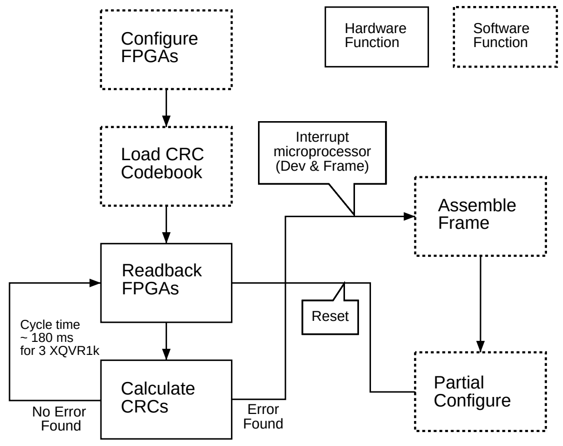


Fig. 4. On-Orbit SEU-Induced Fault Detection and Correction

the FPGA with ‘corrupt’ frames. This stimulates the system to verify that the response to an SEU is correct at the logic and software level. Extensive SEU testing can be done in this way.

B. Detecting and Isolating Permanent Faults

In addition to continuous monitoring for SEU-induced transient faults, readback and partial reconfiguration can be used to detect permanent failures such as opens or shorts within an FPGA. It is desirable to obtain maximum coverage and isolation of hard faults with a minimum number of configurations. Diagnostic configurations must be either stored on-board or up-loaded from a ground station. Those stored on-board must share resources with mission algorithms. A configuration upload requires one pass over a ground station, during which state of health data must be downlinked and control parameters uplinked.

Coverage-optimized designs have been developed to test the FPGAs on orbit [6]. These designs use Built-In Self Test (BIST) so that the design itself contains stimuli and result accumulation circuitry. The designs test the Configurable

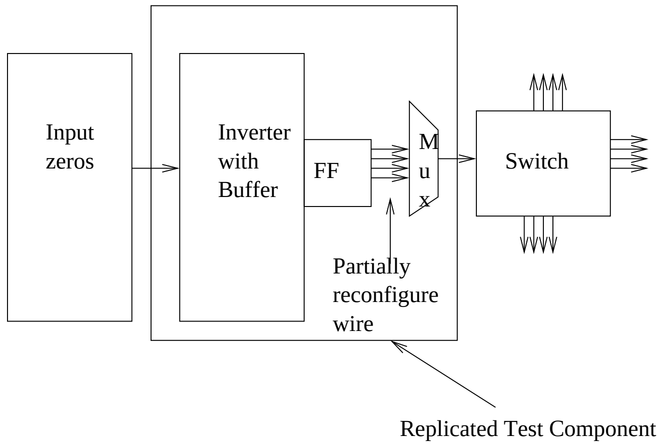


Fig. 5. Partial Reconfiguration to Test Wires

Logic Blocks (CLBs), the Block RAMs (BRAM), and the wires connecting CLBs. CLB tests use a cascade of 34-bit linear feedback shift registers (LFSR) whose inputs are generated by a 6-bit LFSR-based counter. Adjacent 34-bit register outputs are compared, and mismatches ripple through to a CRC module, where the errors are latched. By using two complementary design patterns that alternate placement of the 34-bit registers and 6-bit counters, all the CLBs can be tested using two designs.

For BRAM testing, each location contains its own address in both upper and lower byte, and comparison logic reads out each location, logging mismatches between the bytes.

Single length wires are tested using one design that is repeatedly partially reconfigured. Each CLB has 96 wires, with 24 in each of four directions. Twenty of the wires are part of an output multiplexer. The test procedure first configures the initial test data, filling Column 0 of the FPGA with zero, and all other columns are configured as inverters, with all flipflops initialized to zero. The CLBs are chained together, each using the same output wire of the 96 available wires. Then the clock is stepped once, and the configuration is read back, checking for stuck-at-one faults. The clock is stepped once more, and the configuration is read back to check for stuck-at-zero faults in all the CLBs. The configuration is then partially reconfigured to connect the CLBs using the next wire. This sequence is repeated until all twenty of the wires has been tested. A total of twenty partial reconfigurations and 40 readbacks are required to test 80 output wires of each CLB. The remaining four wires in each direction that are not part of the output multiplexer must be tested with a different design. Figure 5 illustrates this process. For clarity, only four wires are shown on the multiplexers.

C. Limitations to Readback-based Techniques

We note that there are limitations to this technique of configuration readback to detect and correct SEU faults. First, an error-free readback of the configuration bitstream does not guarantee that an SEU did not occur. The FPGA contains hidden state that cannot be read back, and upsets to hidden state can conceivably cause errors in the design without any bitstream errors being detected. Further, SEUs in flip-flop states can occur without disturbing the bitstream. Another limitation relates to the use of look-up tables (LUTs) as memory elements within Configurable Logic Blocks (CLBs). If a LUT is used as a memory element and that memory element is written while the readback operation is taking place, the bitstream will be corrupted on readback. Thus we must either disable the fault manager while running a design that uses LUTs as memory elements or stop the clock to do readback. For similar reasons, the on-chip Block SelectRAM (or BRAM) memory cannot be reliably read back without stopping the clock. Further, the output registers of the BRAMs become corrupted during readback. Thus, for BRAM arrays, error detection and correction must be handled via ECC or checksums since readback of BRAM cannot be reliably performed while the design is running.

Even with these limitations to dynamic reconfiguration on the Virtex V1000 FPGAs, our reconfigurable radio represents a revolutionary advance in space-based processing. First, our choice of high performance SRAM-based FPGAs allows us to perform complex signal processing algorithms in orbit. In contrast, the current state of practice uses slow, expensive, radiation-hardened electronics that is not capable of deployed processing—data must be captured and then downlinked to the ground, with all processing and analysis occurring on the ground. Second, the use of reconfigurable FPGAs allows us to upload new designs, enabling deployment of new algorithms in the reconfigurable radio. The current state of practice uses anti-fuse FPGAs or other fixed logic so that the algorithm is fixed at launch time. Finally, our fault detection and correction scheme exploits the Virtex FPGA's readback and partial reconfiguration features, allowing us to correct the effects of bitstream SEUs on the algorithms while they are running.

III. Single event upset simulation

While it is important to detect and correct SEU-induced configuration bitstream faults on-orbit, it is equally important to understand the effect of faults on representative signal processing algorithms that will be fielded. We are particularly interested in detecting *errors* in signal processing results that might occur as a result of SEU-induced faults in configuration memory. A complementary aspect of our SEU-detection and mitigation project is to induce faults artificially into our designs and study the effects as the circuits function. For a given application design, we can classify configuration memory bits whose upset induces errors as being *sensitive*.

One method for inducing faults is to use ground-based radiation sources as in [1]. We (and others, for example [7], [8]) have put FPGAs under heavy ion (linear accelerator) and proton (cyclotron) radiation and observed SEUs during both static testing (i.e., the device is configured but the design is not executing) and dynamic testing (i.e., the design is executing). While testing has established that these radiation tolerant FPGAs are latchup-immune and SEU sensitive, extensive and repeated testing in the cyclotron is not feasible due to availability and cost.

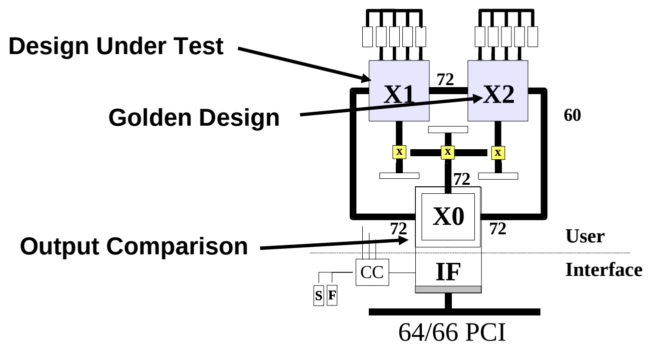


Fig. 6. SEU Simulation on the SLAAC1V

A. The SEU Simulator

To study SEU fault effects on our designs without using the cyclotron, we have developed a bench-testing methodology based on dynamic reconfiguration. We use the simulator to locate sensitive bits in configuration memory: bits whose upset results in errors in application results. Our fault injection tool accurately predicts the behavior of an FPGA design to within 98% [9].

We use an SEU simulator [10] that dynamically reconfigures the FPGA under test with corrupted configurations. Our testbed uses the SLAAC-1V PCI board from USC/ISI [11]. This board has three Xilinx XCV1000 FPGAs, ten 256x36 ZBT SRAMs, and a PCI bus interface. The three FPGAs are connected by a three-port crossbar and share a common clock and reset. In addition, the board includes a configuration controller—a Virtex XCV100 dedicated to configuration loading, partial reconfiguration, and readback.

With this platform, we can load identical designs into the X1 and X2 FPGAs, start the clock, insert artificial configuration upsets, and observe results (Figure 6). As the designs run, we selectively modify the bitstream of the device under test (DUT) to simulate a single-event upset within the configuration bitstream. With a modified bitstream operating on the device, we monitor the effects of the configuration corruption by comparing results produced from the two X1 and X2 FPGAs. By performing a real-time comparision of the two FPGAs on a clock-by-clock basis, the X0 design is able to detect when configuration upsets in the DUT cause the design to deviate from its expected behavior.

By using this simulation technique, we can locate the *sensitive* configuration bits associated for each unique design. A sensitive configuration bit for a particular design is a configuration bit that defines the operation of an FPGA resource used by the design under test. Since each user design will not use the same FPGA resources, the configuration sensitivity will vary from design to design. The purpose of the SEU simulator is to quickly determine the configuration sensitivity of any given user design.

We have obtained SEU simulation results for a variety of user designs and continue to use the simulator to evaluate designs intended for the space-based reconfigurable computer. Table I lists the SEU sensitivity for several synthetic designs.

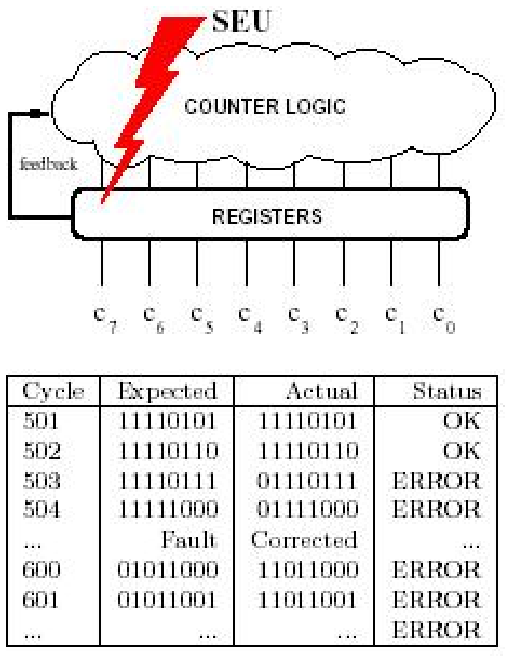
TABLE I
SEU SIMULATOR RESULTS FOR TEST DESIGNS

Design	Logic Slices	Failures	Sensitivity	Normalized Sensitivity
LFSR 18	2178 (15.8%)	667907	1.15%	7.3%
LFSR 36	4356 (31.5%)	137861	2.37%	7.5%
LFSR 54	6534 (47.3%)	208536	3.59%	7.6%
LFSR 72	8712 (63.0%)	279450	4.81%	7.6%
VMULT 18	583 (4.2%)	60929	1.05%	24.9%
VMULT 36	2206 (16.0%)	232239	4.00%	25.0%
VMULT 54	4781 (34.6%)	520747	8.96%	25.9%
VMULT 72	8308 (60.1%)	856802	14.75%	24.5%
MULT 12	144 (1.0%)	13263	0.23%	21.9%
MULT 24	561 (4.1%)	52454	0.90%	22.2%
MULT 36	1249 (9.0%)	122657	2.11%	23.4%
MULT 48	2205 (16.0%)	220197	3.79%	23.8%

The table identifies the size of the design and the number of design failures observed in the SEU simulator. From these results we can calcuate the sensitivity of the design to upsets within the configuration memory. This result is obtained by dividing the number of design failures by the total number of configuration upsets. The `LFSR 72` design, for example, is sensitive to 4.81% of the bits within the configuration bitstream. In addition, this table normalizes the configuration sensitivity by factoring out the effects of area on the sensitivity result. As expected, similar designs of varying sizes exhibit very similar normalized sensitivity.

In addition to configuration sensitivity, our SEU simulator can further categorize the sensitive bits into *persistent* and *non-persistent* [12]. Persistent configuration bits introduce permanent errors in the design state and can only be repaired by both reconfiguring the offending bit and reseting the device. Non-persistent bits, however, result in transient errors that will eventually disappear. After the non-persistent sensitive bit is repaired through traditional configuration scrubbing, the design operates normally and does not require a reset. Persistent bits are most often associated with state and control functions. For example, Figure 7 illustrates error persistence after the high bit of a counter has upset. After cycle 502, the actual counter value never matches the expected result. The design must be reset in order to re-synchronize the counter.

We applied our simulation approach to several additional designs to identify the persistence of the configuration bitstream. These results are shown for four desings in Table II. For each design, the table provides the design size, configuration sensitivity, and persistence ratio. The persistence ratio is the ratio of persistent configuration bits to sensitive configuration bits. As seen in this table, the persistence varies greatly for each design. For our simple feedforward multiplier-adder design, no persistent configuration bits were found. However, the largely sequential nature of the LFSR design exhibits a very large number of persistent configuration bits. The persistent configuration bits ratio is an important parameter that will be used to help the designer select the appropriate SEU design mitigation strategy.



Cycle	Expected	Actual	Status
501	11110101	11110101	OK
502	11110110	11110110	OK
503	11110111	01110111	ERROR
504	11111000	01111000	ERROR
...	Fault	Corrected	...
600	01011000	11011000	ERROR
601	01011001	11011001	ERROR
...	...	...	ERROR

Fig. 7. Errors Induced by Persistent Configuration Bits

TABLE II
SEU SIMULATOR RESULTS FOR TEST DESIGNS

Design	Logic Slices	Sensitivity	Persistence Ratio†
54 Multiply-Add	4781 (39.0%)	8.87%	0%
36 Counter/Adder	36 (.3%)	,09%	9.88%
72 LFSR	8712 (71%)	4.2%	93.9%
LFSR Multiplier		6.4%	15.0%
Filter Preproc.		9.5%	1.2%

† Persistent bits per sensitive configuration bit

An important goal of our simulation is complete coverage of the configuration bitstream, which can only be accomplished through fast, partial reconfiguration of the device under test. We achieve rapid fault injection by using SLAAC-1V's high-speed PCI configuration mode along with the Virtex SelectMAP configuration interface. On our testbed, a single bit can be modified and loaded in 100 μs. The simulator, running on the host with support hardware in X0, operates in a simple loop:

- The simulator corrupts the next bit in the configuration bitstream.
- The simulator partially reconfigures the DUT to load the corrupted frame.
- While the clock is running, the comparator circuitry in X0 checks for output discrepancies between the DUT and "golden" designs.
- The simulator logs discrepancies to a file.
- The simulator corrects the current bit.

This process, shown in Figure 8, takes 214 μs, making it possible to exhaustively test the entire bitstream of 5.8 million bits in 20 minutes.

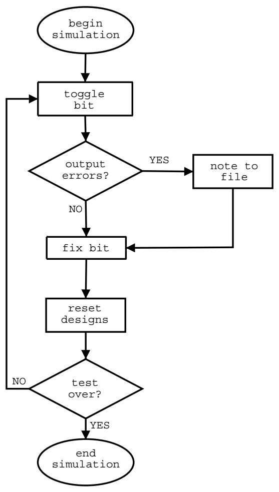


Fig. 8. SEU Fault Injection

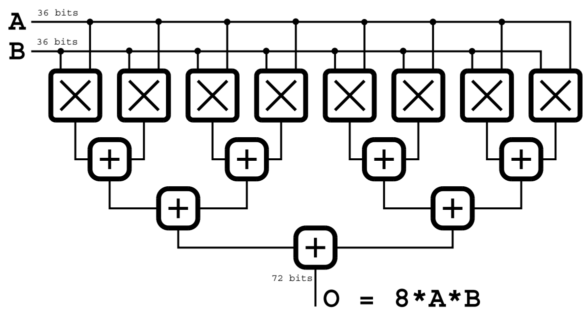


Fig. 9. Pipelined multipy-add tree

By repeated exhaustive tests, it is possible to correlate a single-bit upset in the bitstream with an output error. Such a correlation table was developed for our example designs. High correlation between specific locations in the bit stream and output area helps to characterize the sensitive cross-section of the design. Selective Triple Module Redundancy (TMR) or other mitigation techniques can then be selectively applied to the sensitive cross section.

We have experimented with two design classes that are characteristic of our applications: feed-forward, data-path-dominated designs to assess the impact of SEUs on computation hardware and designs with local feedback (e.g., linear feedback shift registers) to assess the impact of error feedback.

As an example of the data-path-dominated design, Figure 9 shows a pipelined multiply-and-add circuit. In this parallel tree of multipliers and adders, the A and B inputs are each 36-bits and the result is 72-bits.

The LFSR design is shown in Figure 10. It consists of clusters of 20-bit wide LFSRs. Each LFSR cluster contains six LFSRs whose outputs are XOR'ed to form one bit of output. Since the SLAAC-1V testbed provides for 72 output pins, the design contains 72 clusters.

B. SEU Simulator Validation

To assess the accuracy of our SEU simulator, we have also benchmarked the simulator results against dynamic radiation testing. Radiation effects tests have been conducted at the Crocker Nuclear Laboratory, University of California at Davis, USA. This facility houses a 76-inch cyclotron that is used to simulate radiation effects produced by solar and cosmic radiation. During accelerator testing, a proton radiation source was chosen so that there would be approximately one configuration upset per observation. Protons have a low interaction rate, which makes this low rate feasible.

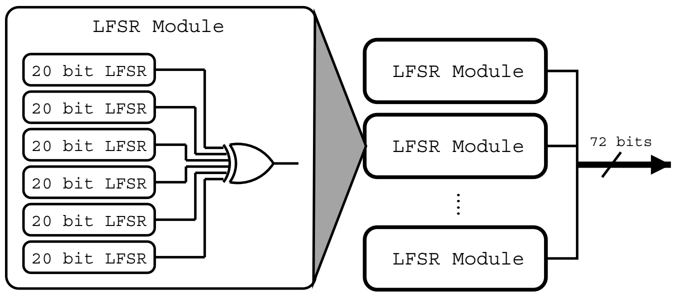


Fig. 10. Linear Feedback Shift Register

For accelerator testing, we use a SLAAC-1V board on a PCI bus extender with the DUT FPGA in a socket so irradiated FPGAs can be exchanged for new FPGAs when needed. Sheets of .75" aluminum shield the FPGA board (except for the DUT) and the PC host. The objective of radiation testing is to operate the designs at speed (up to 20 MHz) in the proton beam while appropriately adjusting the beam's flux so that about one bitstream upset occurrs during each .5-second observation interval. Keeping the SEUs to about one per observation more closely mimics the on-orbit occurrence of SEUs since they are generally isolated events. The test fixture for accelerator testing is diagrammed in Figure 11.

During the test, the output of the DUT is compared with the "golden" part output, and differences are logged along with a timestamp of the occurrence. At the same time, configuration readback is performed at regular intervals, and, when an upset is found, its position is recorded along with a timestamp. If an upset is detected, the FPGA is partially reconfigured to repair the configuration bitstream. If an output error is observed, both designs are reset. The procedure is outlined in Figure 12. Each iteration of the test loop takes about 430μ to complete.

During analysis, output errors that have been predicted by the SEU simulator can be identifed. Analysis of the log data showed a 97.6% correlation between output errors discovered through radiation testing and output errors predicted by the simulator [3]. This validates our bench testing methodology and greatly reduces the number of radiation experiments that must be conducted.

C. Half-Latches

As noted in Section II-C, actual radiation sources can affect state on the FPGA that is not visible in the configuration bit stream. The simulation approach of corrupting the bit stream necessarily can only upset those parts of the FPGA that are defined by configuration bits. While this represents 99.58% of the sensitive cross-section of the FPGA, there is hidden state on the FPGA that is not accessible through the bit stream.

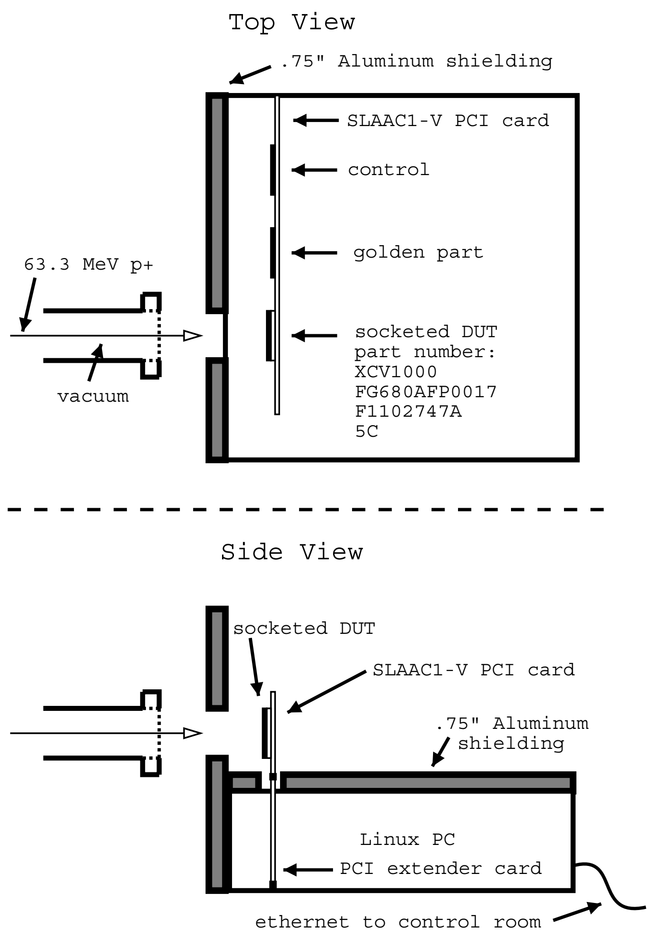


Fig. 11. Accelerator-induced SEU Testing

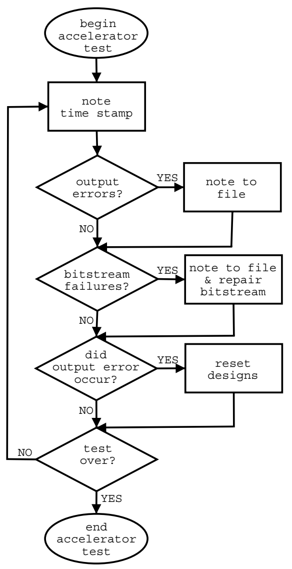


Fig. 12. Accelerator Testing Methodology

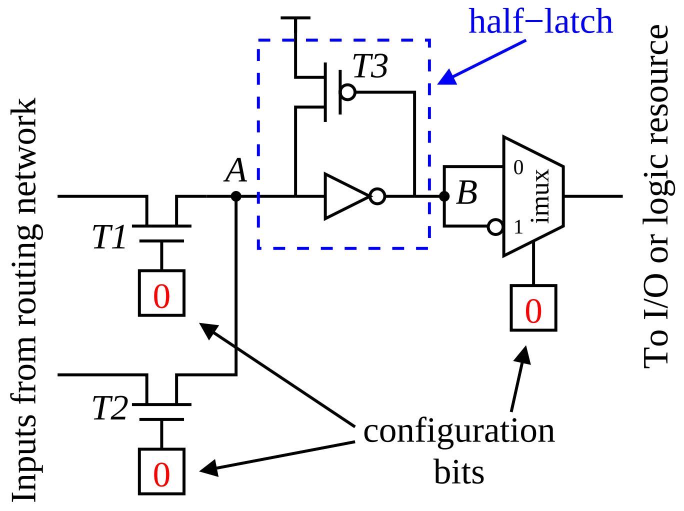


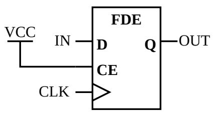
Fig. 13. Half-latch Structure

For instance, the state machines that control read or write of configuration memory and programming of the device are sensitive to SEUs. Upsets in these circuits are often easily detected because the device becomes “unprogrammed.” Other hidden state upsets are less easily diagnosed. Through detailed comparison and analysis of simulated SEUs vs. SEUs observed in the cyclotron, upsets of Virtex FPGA *half-latches* structures have been revealed [13].

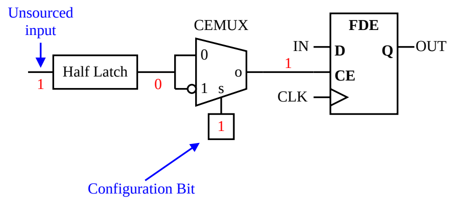
A half-latch in the Virtex FPGA is an internal FPGA resource that efficiently provides constant logic values (one or zero) throughout the device, a savings over the use of look-up tables to store constants. Half-latches are found at the input to logic resources such as Input Output Blocks (IOB), slices, clock resources, and RAM blocks when there areno direct sources for the input, i.e. the inputs are left unconnected. The Xilinx Computer Aided Design (CAD) tools use half-latches frequently to provide constants in circuits. We have found that a large Virtex design can yse hundreds to thousands of half-latches. Half-latches driving input multiplexers are generally critical to design operation if used. Half latches driving LookUp Table (LUT) inputs are not as critical since LUTs are redundantly encoded so that if an unused input attached to a half-latch is inverted, the LUT output is not affected. Figure 13 is a simplified circuit-level representation of a hlaf-latch. The half-latch is the PMOS transistor (T3) and inverter pair between input NMOS transistors from the routing netowrk and the resource input multiplexer. The half-latch holds a “one” value when T1 and T2 are off. However, T3 is a weak pull-up, so that it can be overridden by signals from the routing network when T1 or T2 are on. In this circuit the mux following the half-latch can select between direct output or inverted output (“B” in Figure 13), so that the circuit can supply logic 0 or 1 as needed in the encompassing application circuit. When a Virtex FPGA is fully configured (the configuration memory is updated and a start-up sequence is executed), all half-latches in the device are initialized to the proper state (1 at node A).

The half-latch circuit can experience SEUs, causing the output value to become inverted. The half-latch may recover over time, but this is a stochastic process. Spontaneous recovery to original state was observed during proton testing. However, the only reliable recovery process is to perform a full reconfiguration with the standard start-up sequence. Partial configuration does not restore the half-latch as it does not execute the start-up sequence, and configuration bitstream readback does not detect half-latch state.

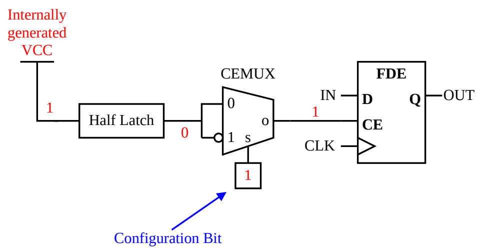
Figure 14 illustrates the impact of half-latch upset. The desired circuit is a flip-flop with clock enable always asserted, as shown in Part (a). The CAD tool chooses a half-latch to supply constant 1 to drive the clock enable (Part (b)). During the start-up sequence associated with full configuration, the half-latch is initialized to zero (Part (c)). When a proton upsets the half-latch, its output is inverted, thus disabling the flip-flop. The upset cannot be detected via readback nor repaired via partial reconfiguration.



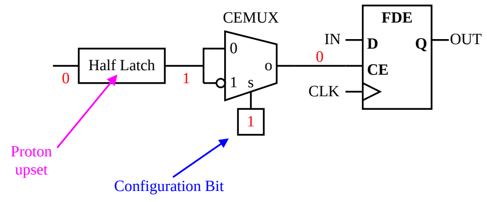
(a) Flip-flop with Clock Enable On



(b) Half-latch Provides Constant 1



(c) Half-latch Initialization



(d) Half-latch SEU

Fig. 14. Half-latch illustration showing the intended circuit, the circuit's usual low-level implementation, the initialization of the half-latch at start-up, and the result of a half-latch SEU.

There are several alternatives to the use of half-latches to generate constants. A constant zero or one can be obtained from external input and then used as the source of all other constants required in the design. Alternatively, a constant can be stored in a look-up table configured as a Read Ony Memory (ROM). Finally, a flip-flop can be used to store a constant. Design mitigation to remove half-latches is best performed automatically rather than by the designer. To this end, we have developed a half-latch removal tool RadDRC that automatically removes half-latches from an application design. The half latches are replaced either by constants from an external source or by LUT ROM constants. Mitigated designs were found to be 100X resistent to failure than unmitigated designs, as observed under Crocker cyclotron testing [13].

IV. Readback and Reconfiguration: Architectural Implications

Our choice of Xilinx FPGAs was driven by the desire to read back the configuration data and partially reconfigure the device. Readback, of course, enables bitstream SEU error detection while partial configuration allows us to repair just the portion of the bitstream that is corrupted as the design runs, avoiding the overhead of stopping the design and then completely reloading the programming data. With the utility of partial configuration and readback for improving design reliability in the space radiation environment come several shortcomings with the current generations of Xilinx SRAM FPGAs.

A. Readback

As pointed out in Section II, it may be difficult to use LUTs as RAMs or shift registers on Xilinx Virtex FPGAs when using configuration readback. As stated in [14], a LUT being used as a RAM or shift register must not be written to as its contents are being read out by the FPGA's configuration circuitry since doing so can corrupt the contents of the LUT. Further, the CRC-based scheme described earlier for detecting errors in bitstream frames must be altered to mask out the contents of LUTs actively used as dynamic storage or a more bit-level comparison must be made, again, masking out regions of LUT state being used as RAM or shift-register storage.

Ideally, the FPGAs themselves could be designed so that this dual access of the LUT memory could be performed without corrupting the data. In a sense, the LUTs could be designed to be something like a true dual-ported memory or a memory that has a second "shadow" memory that can be read out without affecting design operation. Another alternative is to design the readback of LUTs so that their locations in the readback stream are set to zeros when the LUTs are being used in RAM mode. This would allow standard CRC checking to be done to the bitstream without having to mask out some locations and would not require dual-port access to the memories.

Since the volume of customers that actually try to perform readback during device operation is fairly small, an FPGA manufacturer such as Xilinx is not likely to make this sort of change since it would increase the area (and therefore the cost) of the their devices. Thus, we must compensate for the readback limitations at the application design and system levels.

At the design and system levels several approaches can and have been taken to deal with this issue. Many do not want to deal with the complexity of using LUTs as RAMs or shift registers during readback and so they are forced to completely avoid the use of LUT RAMs and they must use flip-flops for creating shift registers. For most of our designs, this is the standard approach since we *do* want to use readback for bitstream error location to help protect the integrity of our designs without having to create design-specific methods for handling readback.

Another approach is to not use readback at all to detect configuration bitstream errors but use built-in self-test techniques to periodically validate that the circuit is still functioning correctly. In this case, if an error is found, the test circuitry signals the configuration control circuitry that a configuration error exists and that a full reconfiguration is needed. This second approach was taken by Ray Andraka [15] when designing the 4096-point FFT used in our space application.

Yet another approach is to carefully design each application so that the columns with the LUT memories can be skipped during readback. This is not very convenient with the original Virtex architecture since using a LUT as a shift register or RAM in a single slice within a CLB column would require that 16 out of the 48 configuration data frames for that CLB column not be read back when trying to detect bitstream SEU errors. Further, if LUTs are being used as RAMs or shift registers in both Slice 0 and Slice 1 of the CLB column, 32 out of the 48 frames cannot be readback without causing the above mentioned problems. For Virtex-II, the situation is better since all of the LUT data for a given CLB column is contained in two configuration data frames, so most of the bitstream data for that column of CLBs can be read back during design execution without disturbing the circuit.

Lastly, it should be possible to schedule design operations and readbacks in such a way that they do not happen simultaneously for LUT resources. In the current approach used in our space payload, the bitstream SEU mitigation process executes asynchronously with regards to the operation of the FPGA designs. If this were changed to be synchronous with design operation and the LUT RAMs were not used continuously, it might be possible to schedule the readback of the LUT RAMs for when they are not being written. Alternatively, the design could be created so that writes to LUTs are inhibited during a readback process, or a portion thereof, to avoid the conflict. This approach might work if portions of the design could be stalled at appropriate times without seriously degrading the design's operation.

Block SelectRAM (BRAM) in Virtex has a similar access conflict during readback plus another problem. During readback, the configuration logic takes over the address lines of the BRAM so a design cannot read from or write to the RAM during readback [14]. An additional problem with BRAM and readback is that readback generally corrupts the internal output register of the BRAM so use of the BRAM after readback can be complicated. A work-around for the output register corruption is possible by modifying the user's design (see [16]), but again, it would be much easier for the FPGA user if the output register's initial value was restored once readback was complete by internal BRAM and/or configuration circuitry. Further, to overcome the RAM access problem during readback, the BRAM would effectively need a third memory port so that the operation could be performed without disrupting design operation.

B. Partial configuration

The main complication with using partial readback to repair designs is the granularity of access to the configuration data. With Virtex and Virtex II, the smallest portion of the bitstream which must be read or written is a frame—a collection of hundreds to thousands of bits. Forcing configuration data operations to manipulate at least a frame of data at a time was probably a compromise between providing the flexibility of partial configuration and the silicon costs of implementing partial configuration. In other words, Xilinx probably did not want to pay the costs for full, bit-level random access to the configuration bitstream data so they picked a larger addressable block size.

As a result, the main complication of using partial configuration to repair designs in the presence of dynamic storage resources (such as BRAMs or LUTs used as either RAMs or shift registers) is that a read-modify-write (RMW) operation really needs to be performed to fix bitstream SEUs. If a configuration bitstream data frame is repaired with the original bitstream data when RAMs or LUT-based shift registers are contained in the design, the contents of these dynamic resources will be overwritten with their original initialization state, likely disturbing the operation of the design. For partial configuration to be usable with the current generations of FPGAs, the current state of the BRAM or LUT must be read and then, either that state must be inserted into the original bitstream frame or the readback bitstream frame must be repaired and written back into the FPGA. The big assumption for this to be successful is that the RMW operation can be done before the contents of the RAM or shift register change—a big issue for many designs.

One solution to this issue (as well as the readback issues mentioned above) is to provide a smaller granularity for read and write accesses to the configuration data. This way only the bits that need to be read or written are touched, making it easier to avoid accesses that might disturb circuit operation.

V. Conclusions

In this work, we show practical methods for exploiting readback and dynamic reconfiguration to enhance reliability of non-radiation-hardened FPGAs. Our methodology enables data and compute intensive signal processing algorithms to fly on satellites and in spacecraft. We have built a complete system that judiciously balances radiation-hardened and radiation-tolerant FPGAs to fulfill its goals—radiation-hardened FPGAs are used to monitor, control, and reconfigure the compute engine, which is built from radiation-tolerant FPGAs. Our simulation methodology frees us from extensive beam testing and gives complete, systematic, repeatable coverage of SEU effects by using an FPGA board in a workstation. Simulation results have been validated using proton radiation. We have suggested several architectural, system, and application methods to overcome the limitations of readback/reconfiguration in the current generation Xilinx Virtex FPGAs.

Acknowledgment

The authors would like to thank Xilinx and, specifically, Carl Carmichael and Joe Fabula for their support of this work. We would also like to thank the University of Southern California's Information Sciences Institute (USC/ISI) for their support of our SEU simulation work. Finally, we like to acknowledge the Department of Energy's funding of this work through the Deployable Adaptive Processing Systems and Cibola Flight Experiment projects.

References

[1] E. Fuller, M. Caffrey, P. Blain, C. Carmichael, N. Khalsa, and A. Salazar, "Radiation test results of the Virtex FPGA and ZBT SRAM for space based reconfigurable computing," in *MAPLD Proceedings*, September 1999.

[2] M. Caffrey, "A space-based reconfigurable radio," in *Proceedings of the International Conference on Engineering of Reconfigurable Systems and Algorithms (ERSA)*, T. P. Plaks and P. M. Athanas, Eds. CSREA Press, June 2002, pp. 49–53.

[3] M. J. Wirthlin, D. E. Johnson, N. H. Rollins, M. P. Caffrey, and P. S. Graham, "The reliability of FPGA circuit designs in the presence of radiation induced configuration upsets," in *Proceedings of the IEEE Symposium on FPGAs for Custom Computing Machines (FCCM '03)*, K. L. Pocek and J. M. Arnold, Eds. IEEE Computer Society Press, April 2003, to Be Published.

[4] IEEE, "IEEE Standard for Space Applications Module, Extended Height Format E Form Factor," in *IEEE 1101.7*, 1994.

[5] C. Carmichael, M. Caffrey, and A. Salazar, "Correcting single-event upsets through Virtex partial configuration," Xilinx Corporation, Tech. Rep., June 1, 2000, xAPP216 (v1.0).

[6] P. Sundararajan, R. Wells, B. Patrie, M. Caffrey, and P. Graham, "Testing FPGA devices in space environment," in *Proceedings of the 5th Annual International Conference on Military and Aerospace Programmable Logic Devices (MAPLD)*, September 2003.

[7] R. Katz, K. LaBel, J. Wang, B. Cronquist, R. Koga, S. Penzin, and G. Swift, "Radiation effects on current field programmable technologies," *IEEE Transactions on Nuclear Science*, vol. 44, no. 6, pp. 1945–1956, December 1997.

[8] R. Katz, J. J. Wang, R. Koga, K. A. LaBel, J. McCollum, R. Brown, R. A. Reed, B. Cronquist, S. Crain, T. Scott, W. Paolini, and B. Sin, "Current radiation issues for programmable elements and devices," *IEEE Transactions on Nuclear Science*, vol. 45, no. 6, pp. 2600–2610, December 1998.

[9] E. Johnson, M. Caffrey, P. Graham, N. Rollins, and M. Wirthlin, "Accelerator validation of an fpga seu simulator," in *IEEE Transactions on Nuclear Science*, vol. 50, December 2003, pp. 2147–2157.

[10] E. Johnson, M. J. Wirthlin, and M. Caffrey, "Single-event upset simulation on an FPGA," in *Proceedings of the International Conference on Engineering of Reconfigurable Systems and Algorithms (ERSA)*, T. P. Plaks and P. M. Athanas, Eds. CSREA Press, June 2002, pp. 68–73.

[11] *SLAAC-1V User VHDL Guide*, USC-ISI East, October 1, 2000, release 0.3.1.

[12] D. E. Johnson, K. S. Morgan, M. J. Wirthlin, M. P. Caffrey, and P. S. Graham, "Persistent errors in sram-based fpgas," in *7th Annual Conference on Military and Aerospace Programmable Logic Devices (MAPLD)*, September 2004.

[13] P. Graham, M. Caffrey, E. Johnson, N. Rollins, and M. Wirthlin, "Seu mitigation for half-latches in xilinx virtex fpgas," in *IEEE Transactions on Nuclear Science*, vol. 50, no. 6, December 2003.

[14] Xilinx, "Virtex configuration - how do I perform Virtex readback?" Xilinx, San Jose, CA, Answers Database Record 8181, May 2001.

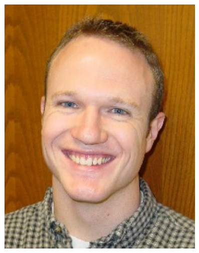
[15] R. Andraka and J. Brady, "Low complexity method for detecting configuration upsets in SRAM-based FPGAs," in *Proceedings of the 5th Annual International Conference on Military and Aerospace Programmable Logic Devices (MAPLD)*, September 2002, B4, To Be Published.

[16] P. S. Graham, "Logical hardware debuggers for fpga-based systems," Ph.D. dissertation, Brigham Young University, Provo, UT, December 2001.

Maya Gokhale is project leader for the Deployable Adaptive Processing Systems and Scalable Reconfigurable Computing projects at Los Alamos National Laboratory. Her research interests include high performance embedded computing and parallel languages and compilers. Dr. Gokhale obtained a Ph.D. in Computer and Information Sciences from the University of Pennsylvania in 1983.

Paul Graham is a member of the technical staff at Los Alamos National Laboratory. His research interests include design automation tools for FPGA-based reconfigurable computing, applications of reconfigurable computing, and FPGA architecture. Dr. Graham obtained an M.S. degree in electrical engineering from Brigham Young University in 1996, and a Ph.D. in Electrical Engineering from Brigham Young University in 2001.

Michael Wirthlin is an assistant Professor at Brigham Young University in Provo, UT. His research interests include tools and applications for FPGA-based reconfigurable computing, power efficiency in FPGA designs, and synthesis techniques for digital systems. Dr. Wirthlin obtained a Ph.D. in electrical engineering from Brigham Young University in 1997.



D. Eric Johnson is a graduate student in the Electrical and Computer Engineering program at Brigham Young University with interests in digital signal processing and reconfigurable computing. He received his B.S. degree from Brigham Young University in 2003.

Nathan Rollins is a graduate student in the Electrical and Computer Engineering program at Brigham Young University with an interest in reconfigurable computing. He received his B.S. degree from Brigham Young University in 2003.